Docket No. AUS920010244US1

HIERARCHICAL CORRELATION OF INTRUSION DETECTION EVENTS

BACKGROUND OF THE INVENTION

1. Technical Field:

The present invention is directed toward presenting network security and intrusion information to a user. More specifically, the present invention provides a scalable method, computer program product, and apparatus for concise display of information relating to a series of security violations.

2. Description of Related Art:

Much of the progress in computer technology in recent years has centered around inter-computer communication. Networks of small-scale computers have all but replaced the bulky mainframe computers of the past. It is less expensive and more efficient for users to share data among single-user workstations and small-scale servers than it is to share computing time on a single mainframe computer.

Increases in connectivity between computers, especially through the Internet, the world's largest and most interconnected computer network, are not without costs. Increased connectivity brings with it an increased likelihood of a security breach or other malevolent activity. Put another way, the more accessible computers become, the more they will be accessed.

It is thus imperative for organizations that rely on networks of computers to have effective security violation detection systems in place to prevent and remedy security compromises. In particular, where many system events that might be categorized as suspicious take place, it is important to be able to sort through a large amount of event data to determine what is actually taking place. When system events are simply "dumped" to a human administrator or user, it is difficult for the human administrator to sort through and make sense of the data.

A number of systems have been developed to address this need. Currently available systems apply pattern-matching techniques to identify automatically particular types of security breaches from system events.

Automated pattern-matching systems that "simulate" reasoning--however much they aid network administrators in interpreting the large volumes of event data produced within a network--are not without limitations. Pattern-matching algorithms are only as effective as they patterns they search for. They have little ability to adapt to situations not envisioned by their designers.

What is needed, then is a system for presenting information about security-related events in a system, that puts the information into a form that a human administrator can readily work with. Further, what is needed is a system that is scalable—that is, one that can be expanded proportionally with the size of the network so as to accommodate networks of various sizes.

SUMMARY OF THE INVENTION

The present invention provides a method, computer program product, and apparatus for presenting data about security-related events that puts the data into a concise form. Events are abstracted into a set data-type identifying the source, target, and category of the event. Sets with common elements are grouped together, and those set-groups having a severity exceeding a threshold are designated "situations." The set-groups and situations are propagated to higher-level systems within a hierarchy of "correlation servers," where they are aggregated to form higher-level set-groups and situations. The hierarchical approach allows for a division of labor among correlation servers in a large network and allows the entire system to be scaled to fit the size of the network.

BRIEF DESCRIPTION OF THE DRAWINGS

The novel features believed characteristic of the invention are set forth in the appended claims. The invention itself, however, as well as a preferred mode of use, further objectives and advantages thereof, will best be understood by reference to the following detailed description of an illustrative embodiment when read in conjunction with the accompanying drawings, wherein:

Figure 1 is a diagram of a networked data processing system in which the present invention may be implemented;

Figure 2 is a block diagram of a server system within the networked data processing system of **Figure 1**;

Figure 3 is a block diagram of a client system within the networked data processing system of **Figure 1**;

Figure 4A is a diagram of security attacks within a preferred embodiment of the present invention;

Figure 4B is a diagram of security attacks within a preferred embodiment of the present invention;

Figure 4C is a diagram of security attacks within a preferred embodiment of the present invention;

Figure 5 is a representation of a security event within a preferred embodiment of the present invention;

Figure 6 is a diagram of a group of security events in a preferred embodiment of the present invention;

Figure 7 is a diagram of groups and situations within a preferred embodiment of the present invention;

Figure 8 is a diagram of groups and situations within a preferred embodiment of the present invention;

Figure 9 is a flowchart representation of a process of generalizing security-related events into situations, in accordance with a preferred embodiment of the present invention;

Figure 10 is a diagram of a hierarchical correlation system in accordance with a preferred embodiment of the present invention;

Figure 11 is a diagram of a delta packet in accordance with a preferred embodiment of the present invention;

Figure 12 is a diagram depicting how two or more delta packets may be combined to yield a new delta packet in accordance with a preferred embodiment of the present invention; and

Figure 13 is a flowchart representation of a process of hierarchically correlating intrusion detection events in accordance with a preferred embodiment of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

With reference now to the figures, **Figure 1** depicts a pictorial representation of a network of data processing systems in which the present invention may be implemented. Network data processing system **100** is a network of computers in which the present invention may be implemented. Network data processing system **100** contains a network **102**, which is the medium used to provide communications links between various devices and computers connected together within network data processing system **100**. Network **102** may include connections, such as wire, wireless communication links, or fiber optic cables.

In the depicted example, server **104** is connected to network **102** along with storage unit **106**. In addition, clients **108**, **110**, and **112** are connected to network **102**. These clients **108**, **110**, and **112** may be, for example, personal computers or network computers. In the depicted example, server **104** provides data, such as boot files, operating system images, and applications to clients **108-112**. Clients **108**, **110**, and **112** are clients to server **104**. Network data processing system **100** may include additional servers, clients, and other devices not shown. In the depicted example, network data processing system **100** is the Internet with network **102** representing a worldwide collection of networks and gateways that use the TCP/IP suite of protocols to communicate with one another. At the heart of the Internet is a backbone of high-speed data communication lines between major nodes or host computers, consisting of thousands of commercial, government, educational and other computer systems that route data and messages. Of course, network data processing system **100** also may be implemented as a number of different types of networks, such as for example, an intranet, a local area network (LAN), or a wide area network (WAN). **Figure 1** is intended as an example, and not as an architectural limitation for the present invention.

Referring to **Figure 2**, a block diagram of a data processing system that may be implemented as a server, such as server **104** in **Figure 1**, is depicted in accordance with a preferred embodiment of the present invention. Data processing system **200** may be a symmetric multiprocessor (SMP) system including a plurality of processors **202** and **204** connected to system bus **206**. Alternatively, a single processor system may be employed. Also connected to system bus **206** is memory controller/cache **208**, which provides an interface to local memory **209**. I/O bus bridge **210** is connected to system bus **206** and provides an interface to I/O bus **212**. Memory controller/cache **208** and I/O bus bridge **210** may be integrated as depicted.

Peripheral component interconnect (PCI) bus bridge **214** connected to I/O bus **212** provides an interface to PCI local bus **216**. A number of modems may be connected to PCI local bus **216**. Typical PCI bus implementations will support four PCI expansion slots or add-in connectors. Communications links to network computers **108-112** in **Figure 1** may be provided through modem **218** and network adapter **220** connected to PCI local bus **216** through add-in boards.

Additional PCI bus bridges **222** and **224** provide interfaces for additional PCI local buses **226** and **228**, from which additional modems or network adapters may be supported. In this manner, data processing system **200** allows connections to multiple network computers. A memory-mapped graphics adapter **230** and hard disk **232** may also be connected to I/O bus **212** as depicted, either directly or indirectly.

Those of ordinary skill in the art will appreciate that the hardware depicted in **Figure 2** may vary. For example, other peripheral devices, such as optical disk drives and the like, also may be used in addition to or in place of the hardware depicted. The depicted example is not meant to imply architectural limitations with respect to the present invention.

The data processing system depicted in **Figure 2** may be, for example, an IBM e-Server pSeries system, a product of International Business Machines Corporation in Armonk, New York, running the Advanced Interactive Executive (AIX) operating system or LINUX operating system.

With reference now to **Figure 3**, a block diagram illustrating a data processing system is depicted in which the present invention may be implemented. Data processing system **300** is an example of a client computer. Data processing system **300** employs a peripheral component interconnect (PCI) local bus architecture. Although the depicted example employs a PCI bus, other bus architectures such as Accelerated Graphics Port (AGP) and Industry Standard Architecture (ISA) may be used. Processor **302** and main memory **304** are connected to PCI local bus **306** through PCI bridge **308**. PCI bridge **308** also may include an integrated memory controller and cache memory for processor **302**. Additional connections to PCI local bus **306** may be made through direct component interconnection or through add-in boards. In the depicted example, local area network (LAN) adapter **310**, SCSI host bus adapter **312**, and expansion bus interface **314** are connected to PCI local bus **306** by direct component connection. In contrast, audio adapter **316**, graphics adapter **318**, and audio/video adapter **319** are connected to PCI local bus **306** by add-in boards inserted into expansion slots. Expansion bus interface **314** provides a connection for a keyboard and mouse adapter **320**, modem **322**, and additional memory **324**. Small computer system interface (SCSI) host bus adapter **312** provides a connection for hard disk drive **326**, tape drive **328**, and CD-ROM drive **330**. Typical PCI local bus implementations will support three or four PCI expansion slots or add-in connectors.

An operating system runs on processor **302** and is used to coordinate and provide control of various components within data processing system **300** in **Figure 3**. The operating system may be a commercially available operating system, such as Windows 2000, which is available from Microsoft Corporation. An object oriented programming system such as Java may run in conjunction with the operating system and provide calls to the operating system from Java programs or applications executing on data processing system **300**. "Java" is a trademark of Sun Microsystems, Inc. Instructions for the operating system, the object-oriented operating system, and applications or programs are located on storage devices, such as hard disk drive **326**, and may be loaded into main memory **304** for execution by processor **302**.

Those of ordinary skill in the art will appreciate that the hardware in **Figure 3** may vary depending on the implementation. Other internal hardware or peripheral devices, such as flash ROM (or equivalent nonvolatile memory) or optical disk drives and the like, may be used in addition to or in place of the hardware depicted in **Figure 3**. Also, the processes of the present invention may be applied to a multiprocessor data processing system.

As another example, data processing system **300** may be a stand-alone system configured to be bootable without relying on some type of network communication interface, whether or not data processing system **300** comprises some type of network communication interface. As a further example, data processing system **300** may be a Personal Digital Assistant (PDA) device, which is configured with ROM and/or flash ROM in order to provide non-volatile memory for storing operating system files and/or user-generated data.

The depicted example in **Figure 3** and above-described examples are not meant to imply architectural limitations. For example, data processing system **300** also may be a notebook computer or hand held computer in addition to taking the form of a PDA. Data processing system **300** also may be a kiosk or a Web appliance.

The present invention provides a method, computer program product, and apparatus for reporting possible security violations in a network data processing system containing several individual client or server computers. The techniques of the present invention allow information about potential security violations to be summarized and presented to a user in a concise and easy to understand format.

Figures 4A-4C depict a number of different scenarios in which attacks (or suspicious activity) directed at a network can occur in a preferred embodiment of the present invention. **Figure 4A** depicts the situation where a single source computer **400** directs a number of attacks **402** toward a single target computer **404**. Attacks **402** may be of a single type, such as in the case of a "denial of service attack," in which target computer **404** would be flooded with electronic mail or other network information from source computer **400**. Alternatively, the attacks may be of different types, such as an attempt to break into a user's account on target computer **404**, coupled with the transmission of a "Trojan horse" program via electronic mail. (A "Trojan horse," much like the famous Trojan horse of classical antiquity, is a computer program that appears useful, but actually contains hidden code with a harmful effect.)

Figure 4B depicts a situation in which a single source computer **406** directs attacks **408** at several computers **410** within the network. **Figure 4C** depicts another situation in which several source computer **412** direct attacks **414** at a single target computer **416**.

One easily understands what is happening within the network when attacks are summarized. That is, if one is told that "Computer A is attacking computers on the network," one knows immediately to do something about "Computer A." If, on the other hand, if one reads a list of all of the possibly suspicious activity in the network, one may experience information overload; one may not realize that a problem exists with "Computer A." This is particularly true when "Computer A" attacks multiple computers, as in **Figure 4B**. The present invention provides a way to summarize reports of suspicious activity in such a way that an administrator operating a computer system within the network can easily identify and correct security problems.

Figure 5 depicts a data representation **500** of a single suspicious event or attack (security attacks being a subclass of event) in accordance with a preferred embodiment of the present invention. When an event occurs, an instance of data representation **500** is generated to record the event. An event can be represented as a triplet (or possibly a grouping of higher cardinality) containing the elements depicted, source computer **502**, target computer **504**, and event category **506**. Source computer **502** and target computer **504** are relatively straightforward in that they represent the source computer from which the event emanates and the target computer of the event. Event category **506** represents the general type of event that has occurred. Possible event categories include, but are not limited to, web server event, e-mail event, Trojan horse, denial of service, virus, network event, authentication failures, and access violations.

Figure 6 is a diagram depicting how multiple identical events may be conceptualized together in a group **600** in a preferred embodiment of the present invention. A group is an aggregation of events in which all of the events have at least one common element.

Any given group has a particular severity level associated with it. The severity level of a given group is a function of 1.) the number of events comprising the group, and 2.) the values of the common elements in the group. For instance, a group representing a large number of very serious attacks (those that attack critical targets or attack in a severe way) will have a high severity level, and a group representing a small number of relatively minor attacks will have a low severity level. Any appropriate formula or function for calculating the severity level of a group may be used.

A situation is a group in which the severity level exceeds a certain pre-determined threshold. The pre-determined threshold may vary depending on how many and which of the three elements all of the events in a situation have in common and based on how serious the particular security events involved are. The threshold for each group may be set by formula or by assigning a fixed number to each group type (e.g., all (1, *, 3) groups have threshold 25). The important thing is for each group type to have an associated threshold.

In this case, group **600** represents that a number of events with source "1," target "2," and event category "3" have occurred. (In this and the examples that follow, numbers are used to represent the values within the event triplets. In an actual implementation, any form of representation could be used; numbers are used here for simplicity.) In the example, the severity level of group **600** is 21 (i.e. 7 points for each event in the group—note again, however, that any formula for calculating a severity level that is reasonably related to the qualitative severity of the events making up the group may be used). If the threshold for this type of group (i.e., one holding (1, 2, 3) events) is set at 18, then group **600** is designated a situation, since group **600**'s severity level of 21 exceeds the threshold 18.

In short, as the terms are used in this document, a group is a collection of similar or identical events. A situation is a group wherein the severity level exceeds a certain predetermined threshold. Situations are reported to an administrator or user as summaries of security attacks.

Figure 7 shows how a situation with a broader scope can be composed from multiple similar events in a preferred embodiment of the present invention. Situation **700** summarizes several events that all have source "1" and target "2." These events include a group of events with event category "3" **702** and a group of events with event category "4" **704**. In response to deriving situation **700** from its underlying events, a user or administrator can be notified simply that "Computer 1 is directing security attacks against Computer 2," through a text message, graphical alert, audio message, or the like.

Figure 8 shows how, in a preferred embodiment of the present invention, a situation **800** with an even broader scope can be generated. Situation **800** represents that source computer "1" is directing various attacks at computers within the network. Situation **800** is generalized from the events underlying situation **802**, which is generalized from event groups **804** and **806**. In addition, situation **800** also includes events from event group **808**, which represents a different target computer that situation **802** and its underlying groups.

TABLE I: Which elements are held in common in each class

Class	Source	Target	Category
Situation 1	X	X	X
Situation 2-1	X	X	
Situation 2-2		X	X
Situation 2-3	X		X
Situation 3-1	X		
Situation 3-2		X	
Situation 3-3			X

Table I is a listing of situation classes in a preferred embodiment of the present invention. Each of these situation classes is defined by which elements of the situation sets are held in common in each of the situations. Situation classes provide a means of categorizing situations according to how many elements are held in common across the events making up the situation. For instance, **Table I** shows that a Situation 2-3 holds events having common source and event categories. A situation 2-3, for example, could contain events (1, 2, 3), (1, 3, 3), (1, 1, 3), (1, 4, 3), etc., since in each one of these events, the source (1) and category (3) are the same.

Situation classes are significant in that they both describe a pattern of attack (e.g., many machines targeting a single machine, or a single machine targeting many machines with the same type of attack, etc.) and a degree of seriousness of the attacks. A Situation 1, for instance, is made up of identical events. For a Situation 1 to even exist, however, a serious level of identical security attacks must exist at a single target, all with the same attack source. This "serious level" of attack may be serious because of the particular type, source or target of the attacks, or it may be serious because of a large number of attack events. Nonetheless, the level of attack at the single target will be very high.

Contrast Situation 1 with Situation 3-3. In Situation 3-3, only the event category is held in common. Thus, attacks may be spread out across many targets. Therefore, the attacks at each target may be on a small scale, but the aggregate number of attacks is large, when all machines in the network are considered.

Figure 9 is a flowchart representation of a process of generalizing events to derive situations, in accordance with a preferred embodiment of the present invention. First a list of events that occurred during a given time period is collected (step **900**). The next event in the list is then considered (step **902**). The event is then placed in all groups in which the event may be classified (e.g., if the event is (1, 2, 3), it may be placed in groups (1, 2, *), (1, *, 3), (*, 2, 3), (1, *, *), (*, 2, *), (*, *, 3), and (1, 2, 3)) (step **904**). If there are more events in the list (step **906**), the process cycles to step **902** to process the additional events. If not, then the thresholds and severity levels for each of the groups is ascertained (e.g., calculated or looked up) (step **908**). Finally, groups that exceed their respective pre-determined thresholds are displayed as situations to a user or administrator (step **910**).

The process of aggregation and correlation as described above may be executed within a single server, such as server **104** of **Figure 1**. As the network increases in size and population, however, the workload on that single server will increase. As the network enlarges, at some point the workload will become so high that the task becomes intractable. It then becomes important to be able to scale the computing power devoted to aggregation and correlation to match the workload.

Figure 10 is a diagram of a hierarchical correlation system in accordance with a preferred embodiment of the present invention. The distributed computer system depicted in **Figure 10** is populated with groups of computers **1000**, **1002**, and **1004**. Each of these groups is monitored by correlation servers **1006**, **1014**, and **1022**. For instance, computer group **1000** is monitored by correlation server **1006**. Correlation servers **1006**, **1014**, and **1022** each monitor their respective computer groups and aggregate groups of events, according to the technique taught earlier in this document.

Each of these event groups, as was earlier noted, has a severity level associated with it. Over time, as the network is monitored and the severity level periodically re-measured, the severity levels may increase or decrease. The change in the severity level for a given group is the "delta" severity of that group (the Greek letter Δ or "delta" is customarily used to denote the change in a quantity).

While the term "delta severity" is used to denote the data which summarizes the severity status of a situation, the reader should appreciate that other data summarizing the situation could be used. The point is that the higher level correlation server should not be getting the raw data to recorrelate, but rather the results from a lower level correlation. Thus, the term "propagating the delta severity" should be interpreted to emcompass other data messages which summarize a situation severity from a lower level to a higher level correlation server. A change from a prior situation severity level is simply the preferred embodiment of the invention.

Correlation server **1006** calculates the delta severity for each group each time the severity is re-calculated. If the severity has not been calculated before, the delta severity is defined to be the current severity. That is, it is the delta severity with respect to a zero severity level. In addition to including the delta severity in the delta packet, all other changes to the group are packaged as well. For example, over time the group may grow to include additional sources and attack destinations. These additional sources and destinations are conveyed in the delta packet.

Correlation creates delta packet **1008** from its calculations of delta severity. Delta packet **1008** contains entries corresponding to all of the groups having non-zero delta severity. A diagram of an example delta packet is provided in **Figure 11**.

Delta packet **1008** is transmitted to correlation server **1010**, which combines delta packet **1008** with delta packet **1012** to form a new delta packet **1016**. Delta packet **1012** is generated by correlation server **1014** in the same way as delta packet **1008**. **Figure 12** depicts one way in which delta packets may be combined.

Correlation server **1010** transmits delta packet **1016** to correlation server **1018**, the highest-level correlation server in the hierarchy. There, delta packet **1016** is combined with delta packet **1020** from correlation server **1022** to form a final delta packet (not shown). This final delta packet is used to update the severity levels for all groups with non-zero delta severity at the top level of the hierarchy.

The first time that delta packets are propagated up the hierarchy, correlation server **1018** will, of course, reflect a zero severity for all possible groups. Thus, the first final delta packet to be assembled will contain the initial severity values for the entire network.

Groups at top-level may be promoted to situations by determining whether the top-level groups' severity values exceed predetermined thresholds. This step is, thus, no different than in the case involving a single correlation server. The situations can then be presented to a human operator.

Figure 11 is a diagram of a delta packet **1100** in accordance with a preferred embodiment of the present invention. Delta packet **1100** contains two data fields. Data field **1102** stores the identities of the groups whose severity levels have changed. Data field **1104** stores the delta severities associated with these groups.

Figure 12 is a diagram depicting how two or more delta packets may be combined to yield a new delta packet in accordance with a preferred embodiment of the present invention. Delta packet **1200** is combined with delta packet **1202**. For each group that is common to both delta packet **1200** and delta packet **1202**, the delta severities are merely added. The other groups and their delta severities are simply appended to new delta packet **1204**, and those groups whose combined delta severities are zero are simply removed from new delta packet **1204**. Note that simple addition of delta severities is not the only way to combine delta packets. Other operations, such as taking an arithmetic or geometric mean, are acceptable as well. Other elements, such as lists of sources and destinations may also be combined.

Figure 13 is a flowchart representation of a process of hierarchically correlating intrusion detection events in accordance with a preferred embodiment of the present invention.

First the events are aggregated into groups as described in **Figure 9** (step **1300**). Next, the delta severities for each group are calculated (step **1302**). Then, the non-zero delta severities and their corresponding groups are propagated up as delta packets to the next level in the hierarchy (step **1304**). At the next level, all of the received delta packets are combined to form a new delta packet (step **1306**). If the current level is not the highest level in the hierarchy (step **1308**), then the process cycles to step **1304** to propagate the new delta packet up to the next level. When the highest level is finally reached, the severity levels stored within the top-level system are updated according to the final (cumulative) delta packet now in memory (step **1310**). Finally, those groups with exceeded severity thresholds are reported to a human operator as situations (step **1312**).

It is important to note that while the present invention has been described in the context of a fully functioning data processing system, those of ordinary skill in the art will appreciate that the processes of the present invention are capable of being distributed in the form of a computer readable medium of instructions and a variety of forms and that the present invention applies equally regardless of the particular type of signal bearing media actually used to carry out the distribution. Examples of computer readable media include recordable-type media, such as a floppy disk, a hard disk drive, a RAM, CD-ROMs, DVD-ROMs, and transmission-type media, such as digital and analog communications links, wired or wireless communications links using transmission forms, such as, for example, radio frequency and light wave transmissions. The computer readable media may take the form of coded formats that are decoded for actual use in a particular data processing system.

The description of the present invention has been presented for purposes of illustration and description, and is not intended to be exhaustive or limited to the invention in the form disclosed. Many modifications and variations will be apparent to those of ordinary skill in the art. The embodiment was chosen and described in order to best explain the principles of the invention, the practical application, and to enable others of ordinary skill in the art to understand the invention for various embodiments with various modifications as are suited to the particular use contemplated.